Dr. Annegret Junker & Fabrizio Lazzaretti

Domain-driven API Design

Architektin bei codecentric

interessiert Microservices, DDD, API-Design, Event-Design und wie man schneller von der Spezifikation zum Code kommt

Architekt bei Wavestone

interessiert API-Design, Event-Design, Microservices und wie man das Leben eines Entwicklers leichter macht

APIs zum Gruseln

Eine API direkt aus der Datenbank

```
BusinessPartner:
  description: An existing business partner
  properties:
    CLIENT:
      type: CLNT
    PARTNER:
      type: string
      minLength: 10
      maxLength: 10
    TYPE:
      type: string
      minLength: 1
      maxLength: 1
    BPKIND:
      type: string
      minLength: 4
      maxLength: 4
    ...
```

Kunde, Lieferant, Vermarktungspartner, …?

Definiert den Typ – aber es ist nicht ersichtlich wie PARTNER und BPKIND zusammenwirken

Eine API formuliert nur aus der Sicht des Backends

```
BusinessPartner:
  description: Business partner of the company
  properties:
    tenant:
      description: Tenant number in a multi-tenancy environment
      type: string
…
    partnerNumber:
      description: Number of the business partner
      type: string
    ...
```

Kunde kann nur über Kombination aus `tenant` und `partnerNumber` identifiziert werden. Ein Zugriff über einen Unique Identifier ist nicht möglich.

Eine pur technische API

```
/customers/guids:
  get:
    description: Delivers the GUID of a customer by its tenant and business partner number
    operationId: getCustomerByPartnerNumberAndTenant
    parameters:
…
    responses:
      200:
        description: Successful operation
        content:
          application/json:
            schema:
              type: string
              format: uuid
```

Das kann nicht wirklich die Lösung sein.

Eine API aus dem Elfenbein-Turm

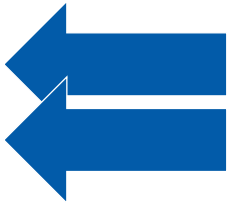
```
/products:
  description: Delivers back all products out of the catalog
  responses:
    200:
      description: Successful operation
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/Product'
```

Es kann Firmen geben, wo der Katalog so kurz ist,
das es funktioniert. Aber in den meisten Fällen wird es nicht funktionieren.
Technische Lösungen wie lazy loading oder paging dürfen auch in geschäftlich-getriebenen APIs nicht fehlen.

Die überladende Welt-Herrschafts-API

```
paths:
  /customers:
    ...
  /customers/{customer-id}:
    ...
  /customers/{customer-id}/addresses:
    ...
  /customers/{customer-id}/addresses/{address-id}:
    ...
  /customers/{customer-id}/accounts:
    ...
  /customers/{customer-id}/account/{account-id}:
    ...
  /customers/{customer-id}/account/{account-id}/transactions:
    ...
```

Kann noch OK sein – hängt vom Kontext ab.



Anderer Kontext und nicht mehr OK.

Definitiv ein anderer Kontext und nicht mehr OK.

APIs ohne Dokumentation und mit unerwarteten Verhalten

```
Customer:
  description: Customer of the company
  properties:
    name:
      type: string
    middleName:
      description: Usual the middle name of a person, in
Spanish speaking countries second family name
      type: string
    firstName:
      type: string
```

Nicht üblich und nicht verständlich

```
{
  "Customers": [
    {
      "name": "Customer not found"
    }
  ]
}
```

Fehler über 200: Nicht üblich und nicht verständlich

Warum passieren diese Fehler

- Nicht Beachten von Standards
  - Lieferung von Fehlern in 200 oder Liefern von Fehlern unter der falschen 4xx
  - Liefern von einfachem Text obwohl application/json oder andere Fomate definiert wurden

- Unterschätzen der Geschäftsrelevanz und Überschätzen der technischen Implementierung
- Überschätzen der Geschäftsrelevanz und Unterschätzen der technischen Relevanz

API Design first
vs.
API Code first

API Design first vs. API Code first

API Design FIRST

- APIs werden geschrieben und Code wird generiert und später verfeinert

- Frühes Einbeziehen aller Stakeholder
- API-Spezifikation wird geschrieben
- Implementierung von Client und Server kann parallelisiert werden
- Nachhaltiges Benutzen der Business-Sprache im Code

- Warten bis alle Stakeholder der API-Spezifikation zugestimmt haben

API Code FIRST

- Code wird geschrieben und APIs werden aus dem Implementierungs-Code generiert

- Kurze Time-to-Market
- Passt zu Proof-of-Concepts (PoCs) und Rapid Prototyping

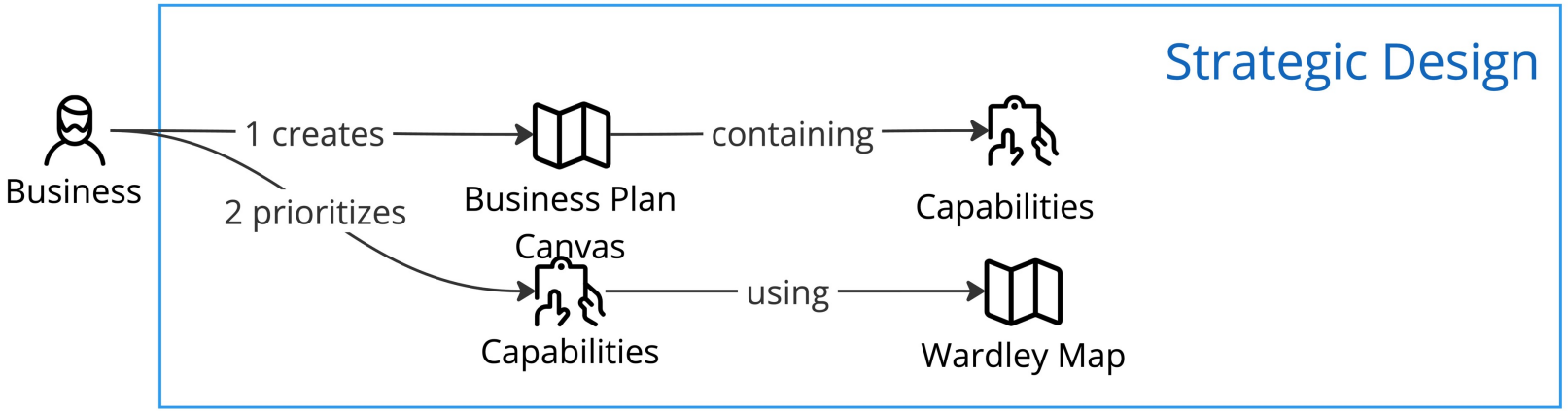
- Stakeholder, Tester, Technische Editoren usw. werden nur unzureichend einbezogen

API Design first vs. API Code first

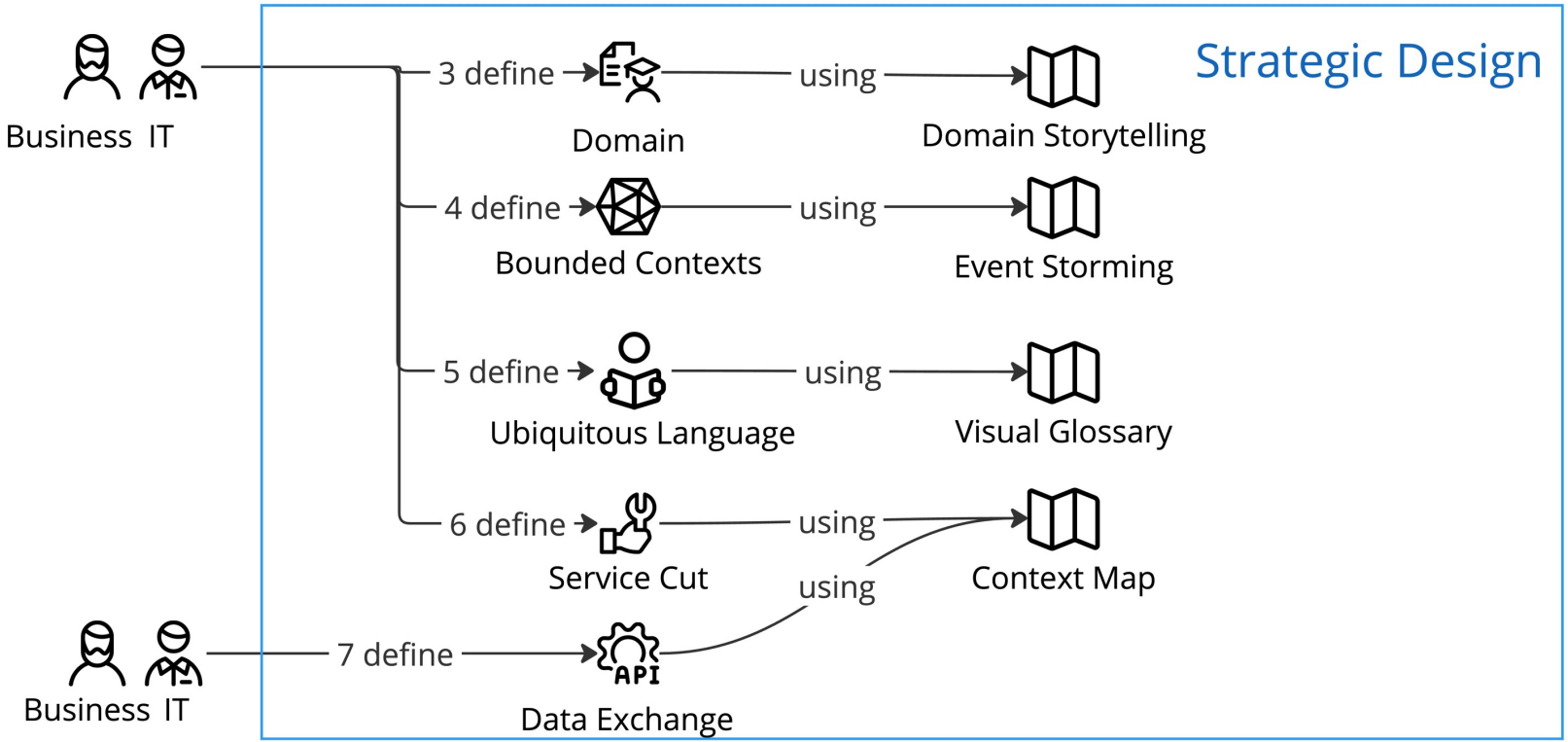
API Design FIRST	API Code FIRST
• APIs werden geschrieben und Code wird generiert und später verfeinert	• Code wird geschrieben und APIs werden aus dem Implementierungs-Code generiert
• Frühes Einbeziehen aller Stakeholder • API-Spezifikation wird geschrieben • Implementierung von Client und Server kann parallelisiert werden • Nachhaltiges Benutzen der Business-Sprache im Code	• Kurze Time-to-Market • Passt zu Proof-of-Concepts (PoCs) und Rapid Prototyping
• Warten bis alle Stakeholder der API-Spezifikation zugestimmt haben	• Stakeholder, Tester, Technische Editoren usw. werden nur unzureichend einbezogen

Synergetic Blueprint

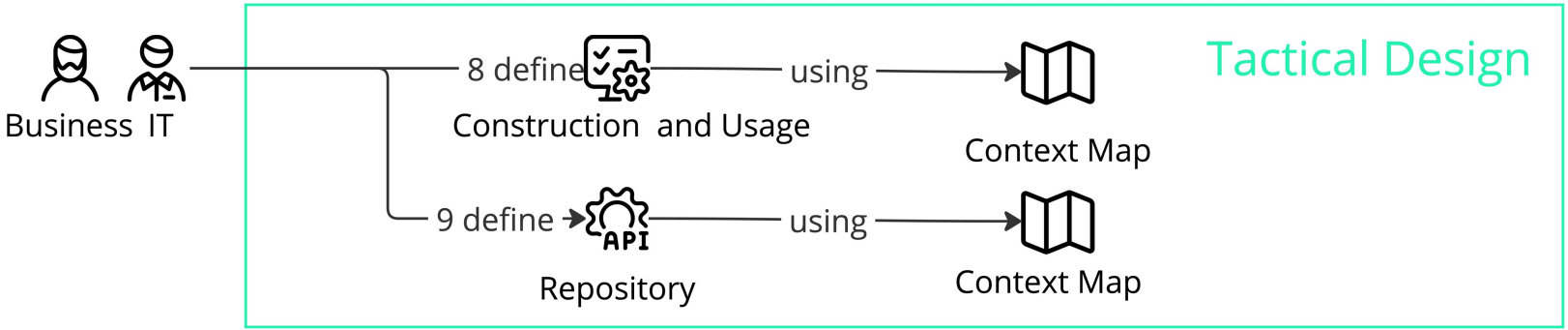
Synergetic Blueprint



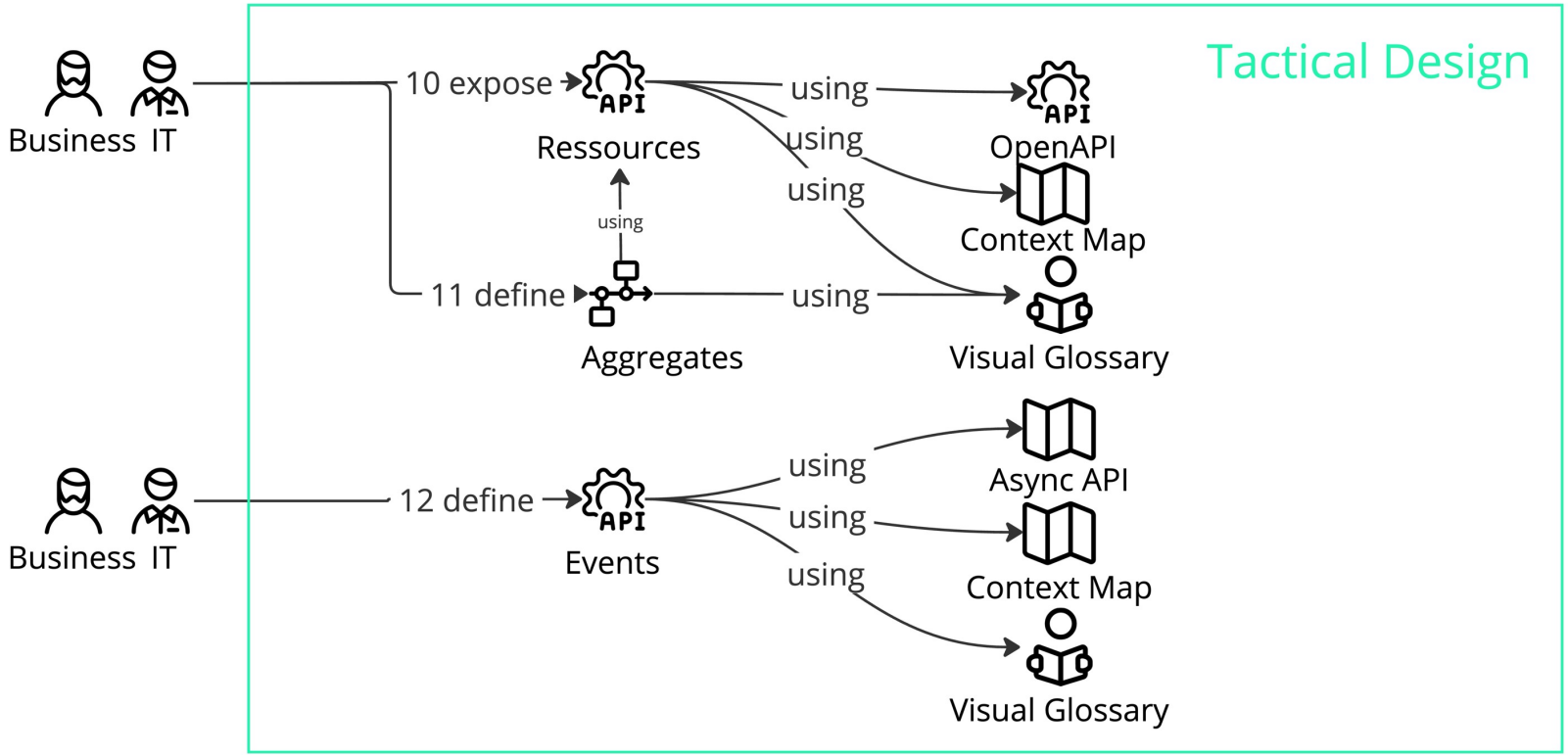
Synergetic Blueprint



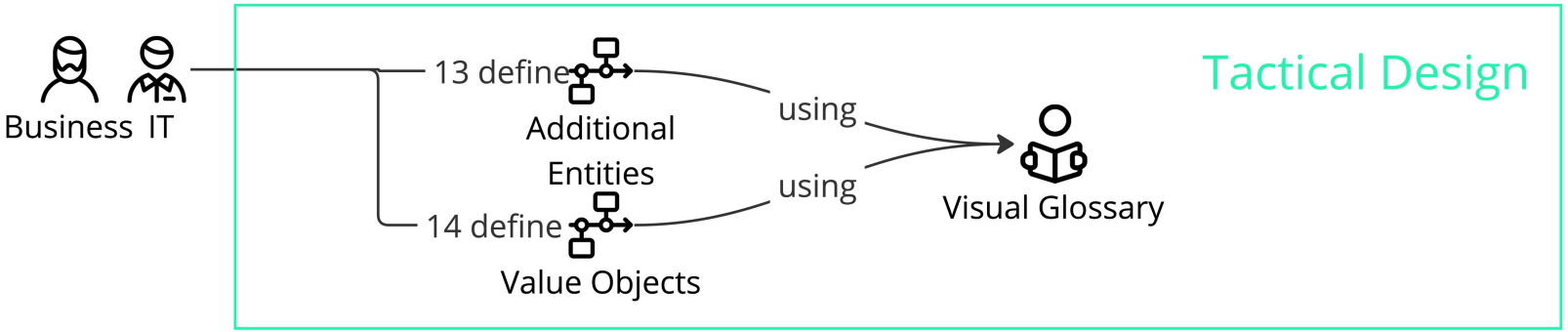
Synergetic Blueprint



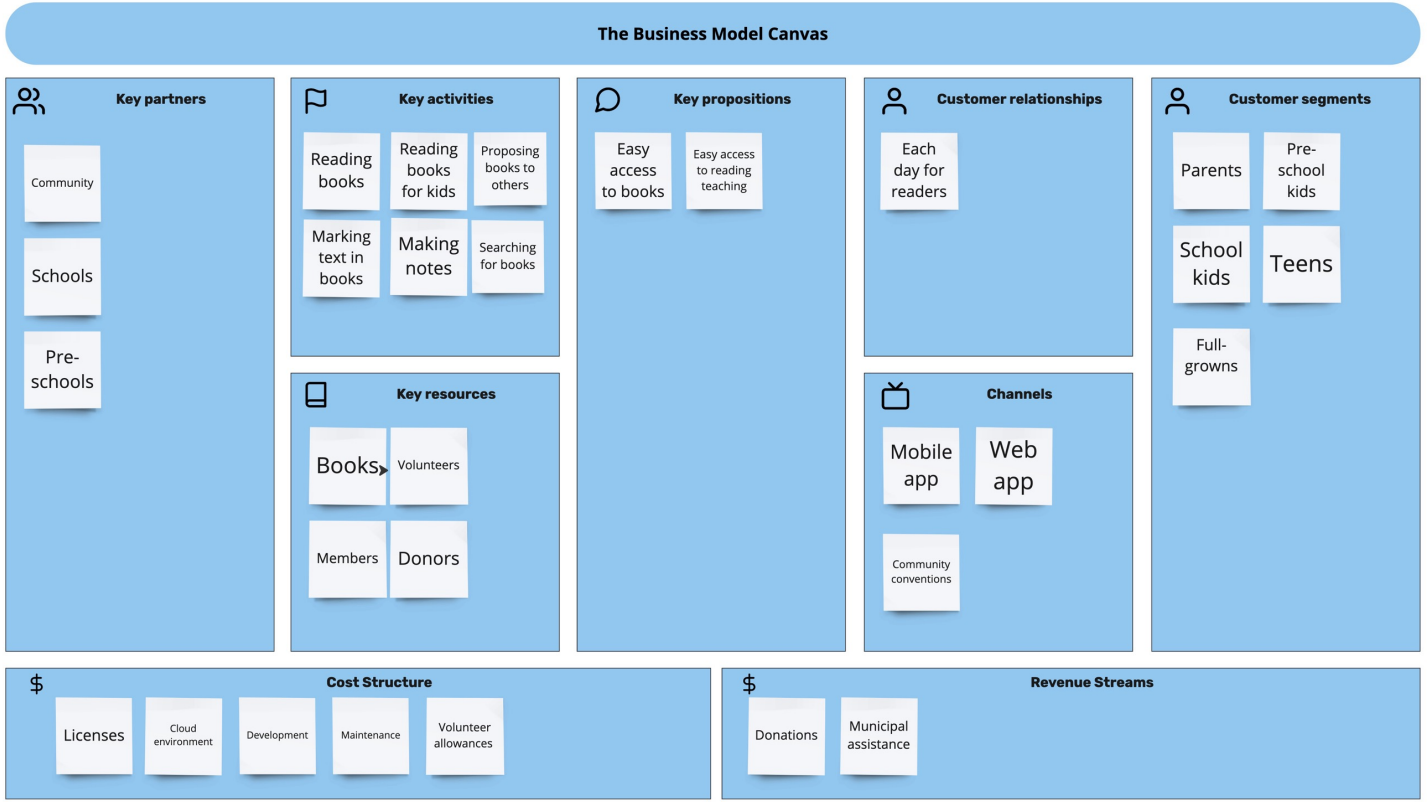
Synergetic Blueprint



Synergetic Blueprint

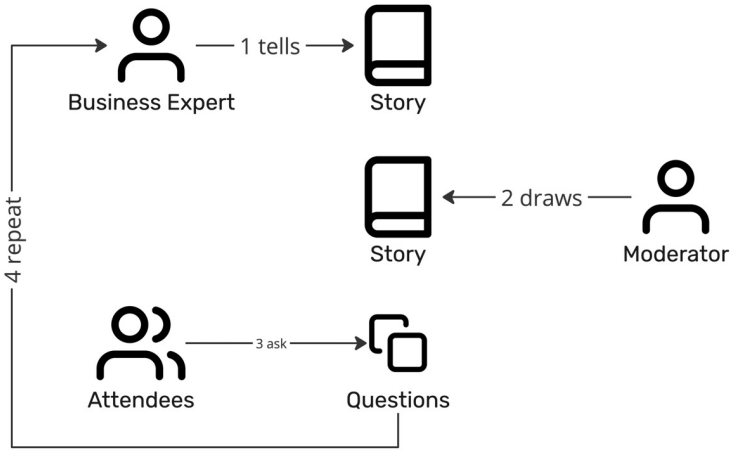


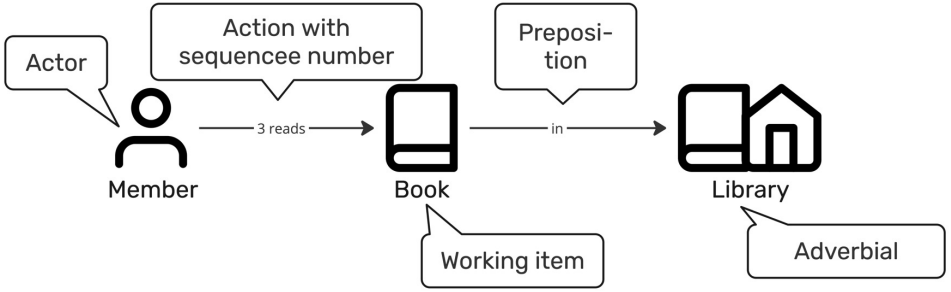
Business Model Canvas



Source: Strategyzer AG | License: CC By-SA 3.0

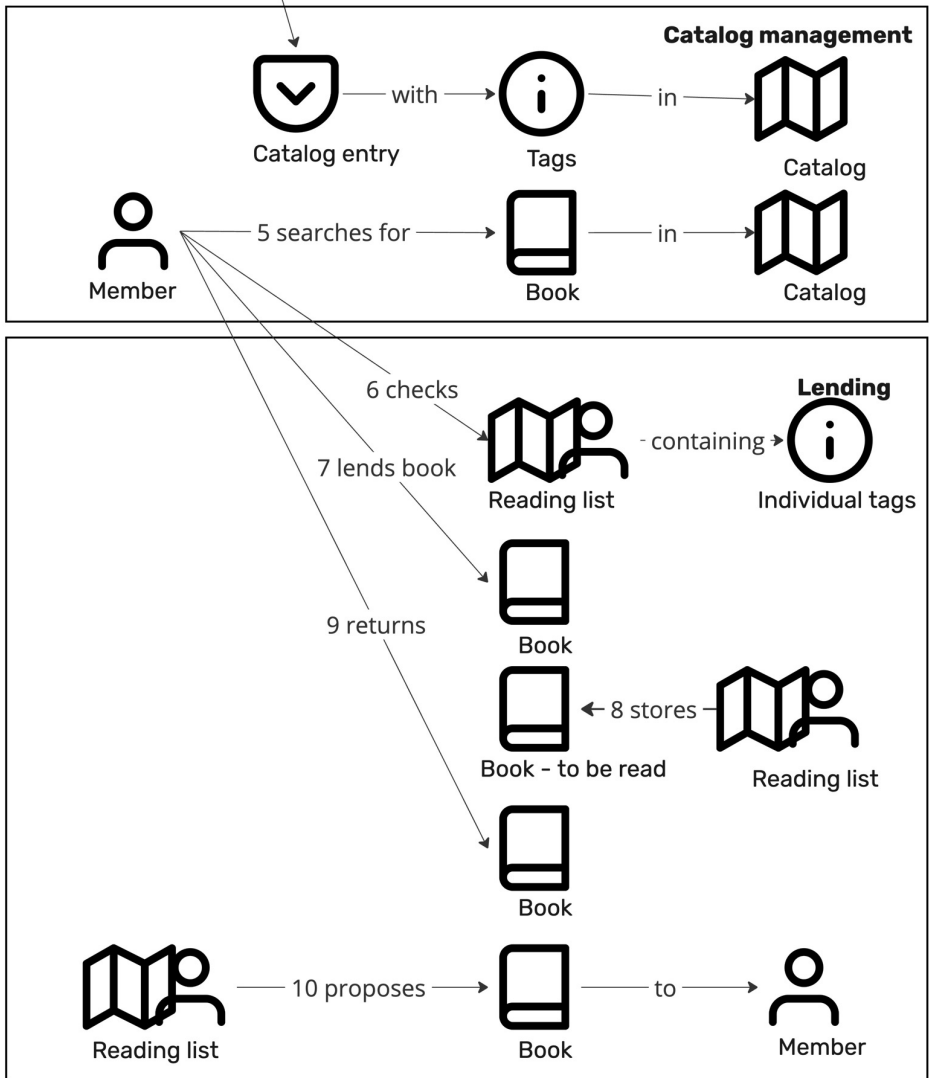
Domain Storytelling





Stefan Hofer, Henning Schwentner: Domain Storytelling

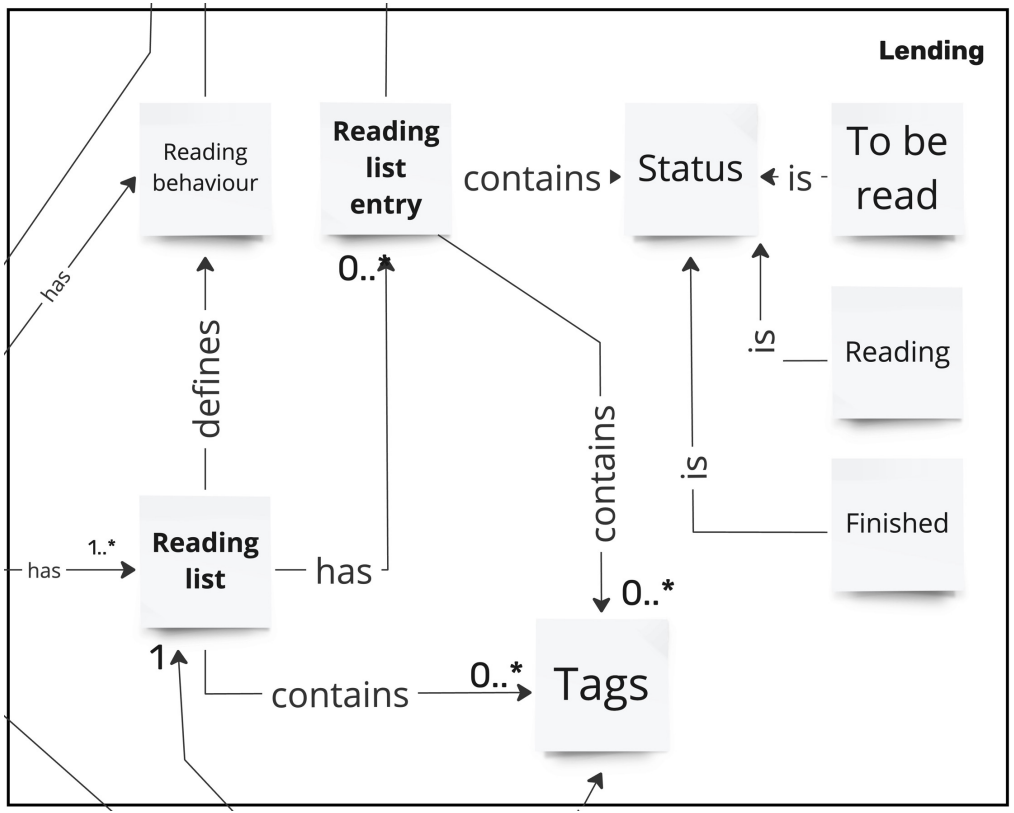
Domain Storytelling
Beispiel Online Bibliothek



codecentric WAVESTONE

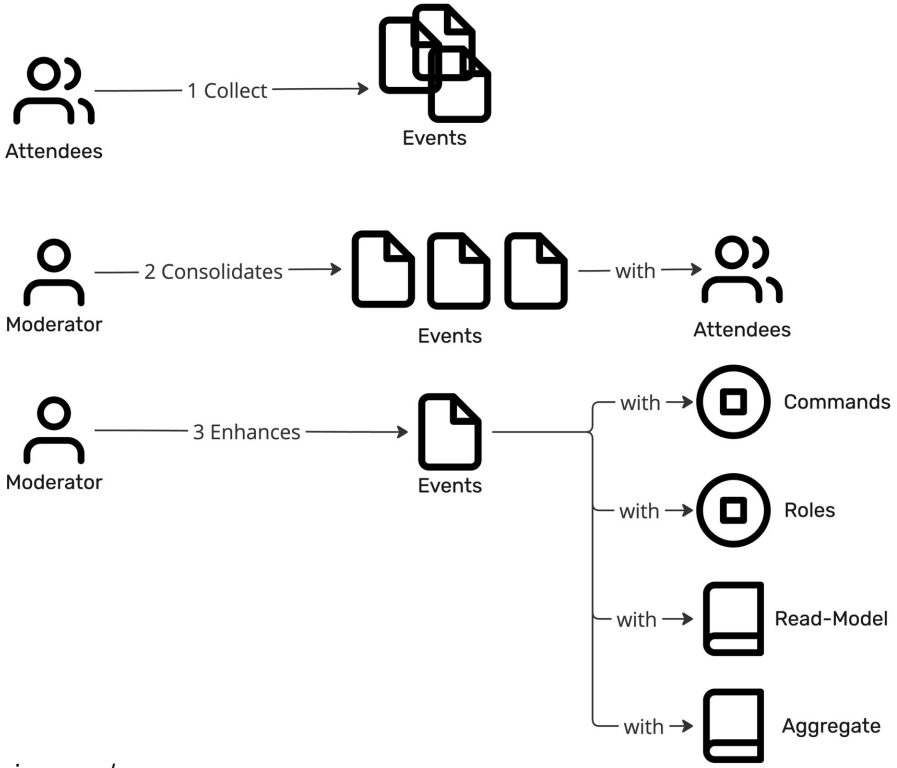
Visual Glossary

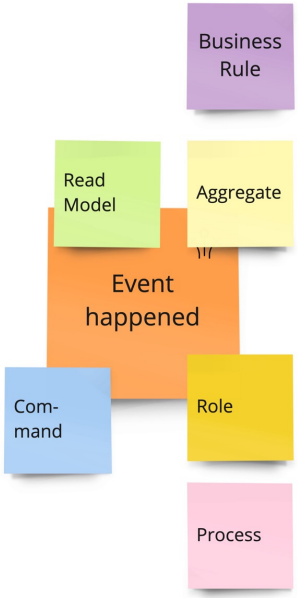
Erläuterung der verwendeten Begriffe durch ihre Beziehungen zueinander



Stefan Zörner: Softwarearchitekturen dokumentieren

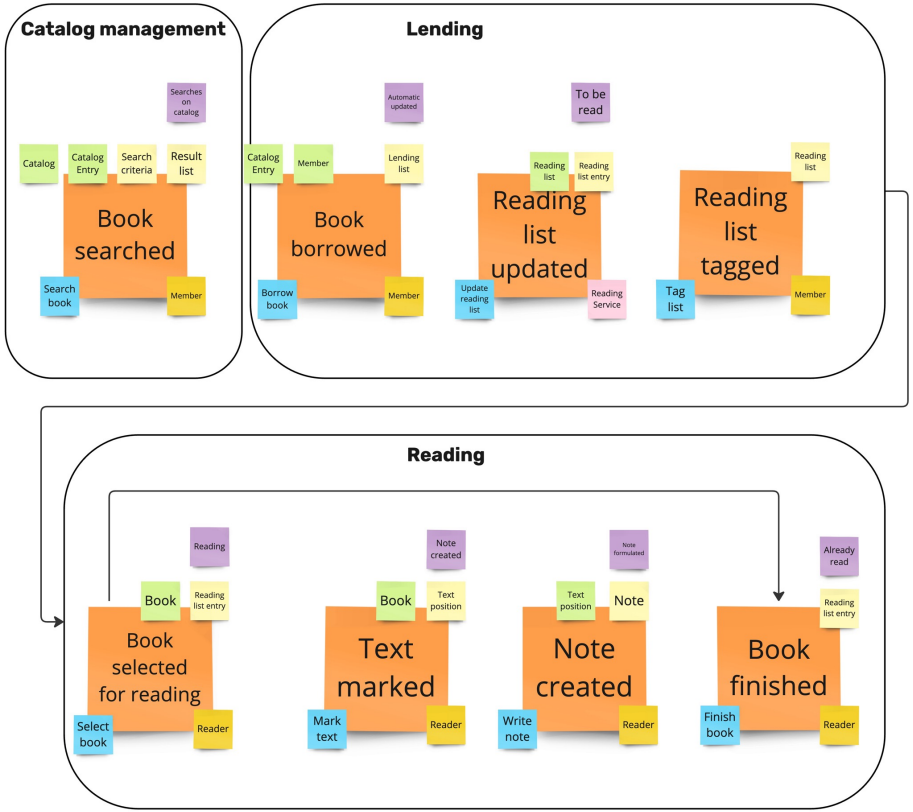
Event Storming



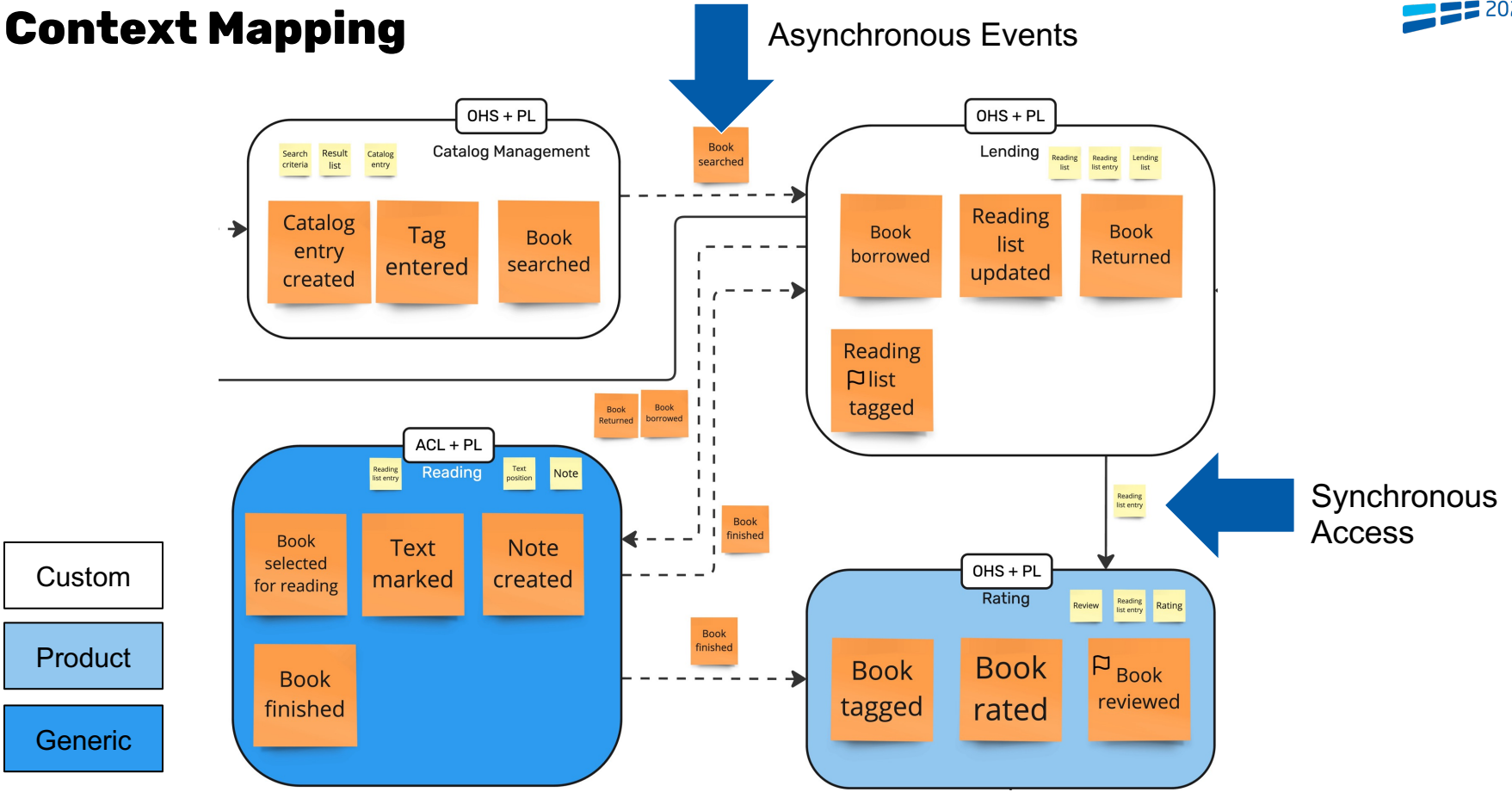


https://www.eventstorming.com/

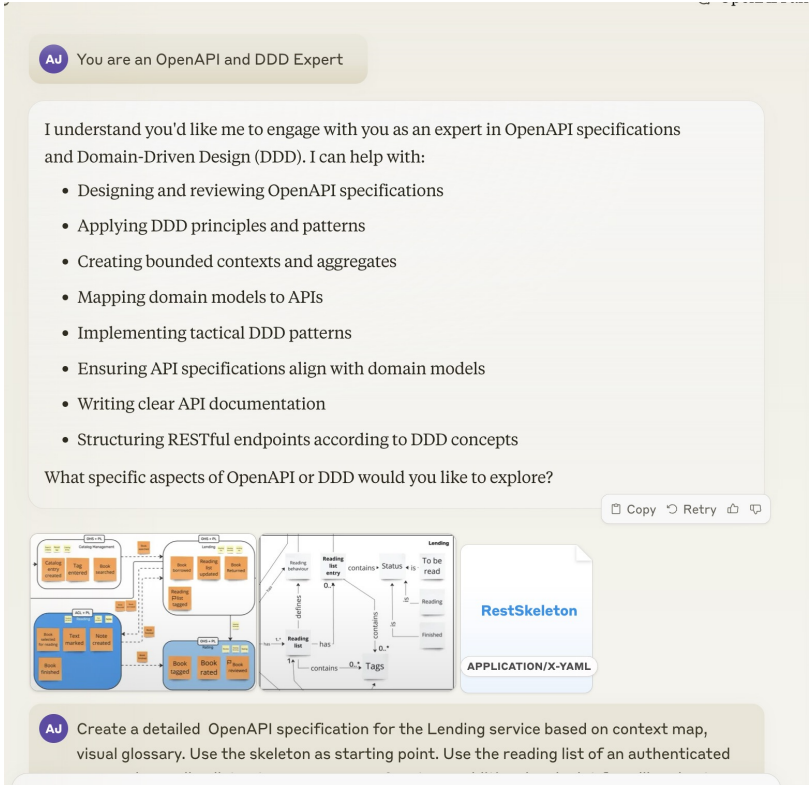
Event Storming



Context Mapping



API Design
Erster Entwurf mit AI



https://github.com/Grinseteddy/DomainDrivenApiDesign/blob/main/IT-Tage2024/ExampleAPI/ChatClaude.md

https://github.com/Grinseteddy/DomainDrivenApiDesign/blob/main/IT-Tage2024/ExampleAPI/lending-api_Claude.yaml

API Design Bearbeitet

```yaml
ReadingListEntry:
  description: Entry of a reading list of a certain user
  type: object
  allOf:
    - $ref: '#/components/schemas/ReadingListEntryCreate'
  required:
    - id
    - status
    - createdAt
  properties:
    id: <4 keys>
    createdAt: <4 keys>
    updatedAt: <4 keys>
```

https://github.com/Grinseteddy/DomainDrivenApiDesign/blob/main/IT-Tage2024/ExampleAPI/lending-api_Edited.yaml

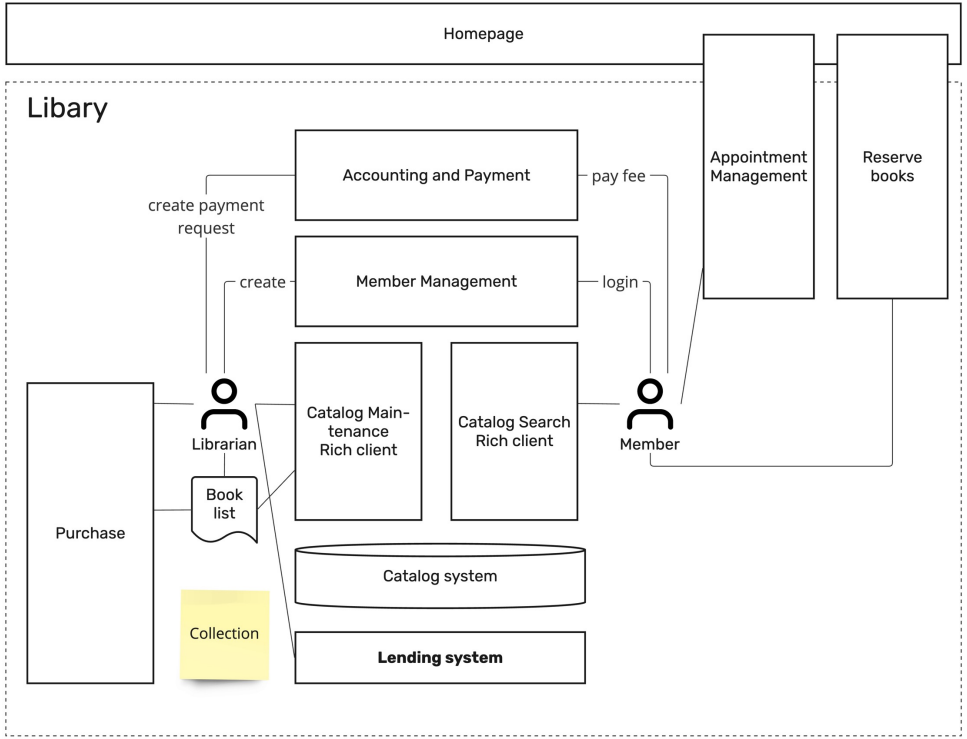
Event Design

```yaml
channels:
  readingList/created:
    address: "lending/readingList.created"
    messages:
      readingListCreated:
        $ref: '#/components/messages/readingListCreated'

  readingList/updated:
    address: "lending/readingList.updated"
    messages:
      readingListUpdated:
        $ref: '#/components/messages/readingListUpdated'

  book/borrowed:
    address: "lending/book.borrowed"
    messages:
```

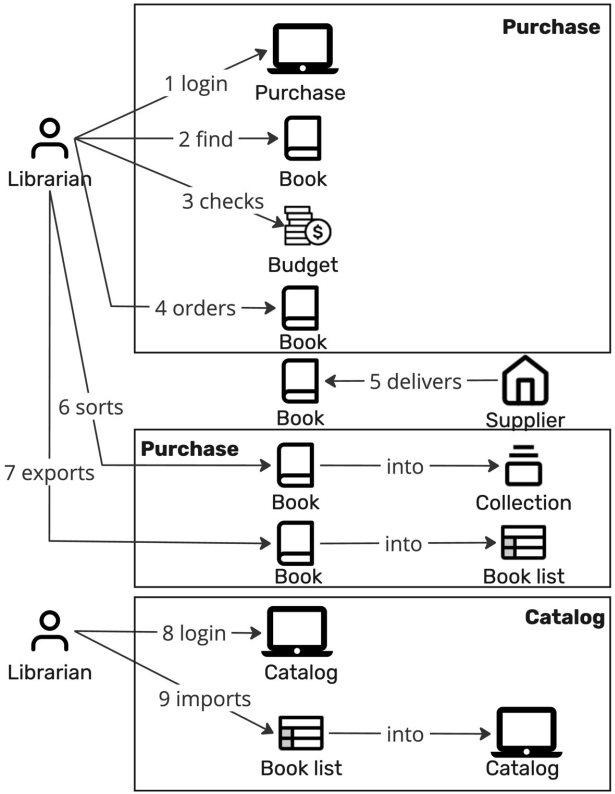
https://github.com/Grinseteddy/DomainDrivenApiDesign/blob/main/IT-Tage2024/ExampleAPI/ClaudeAsyncApiLending.yaml

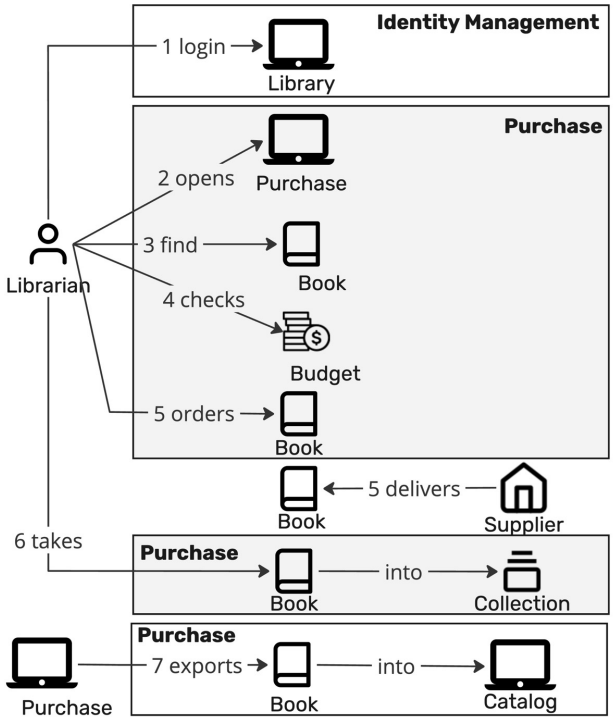
Working in a Brownfield

Analyse jetziges System

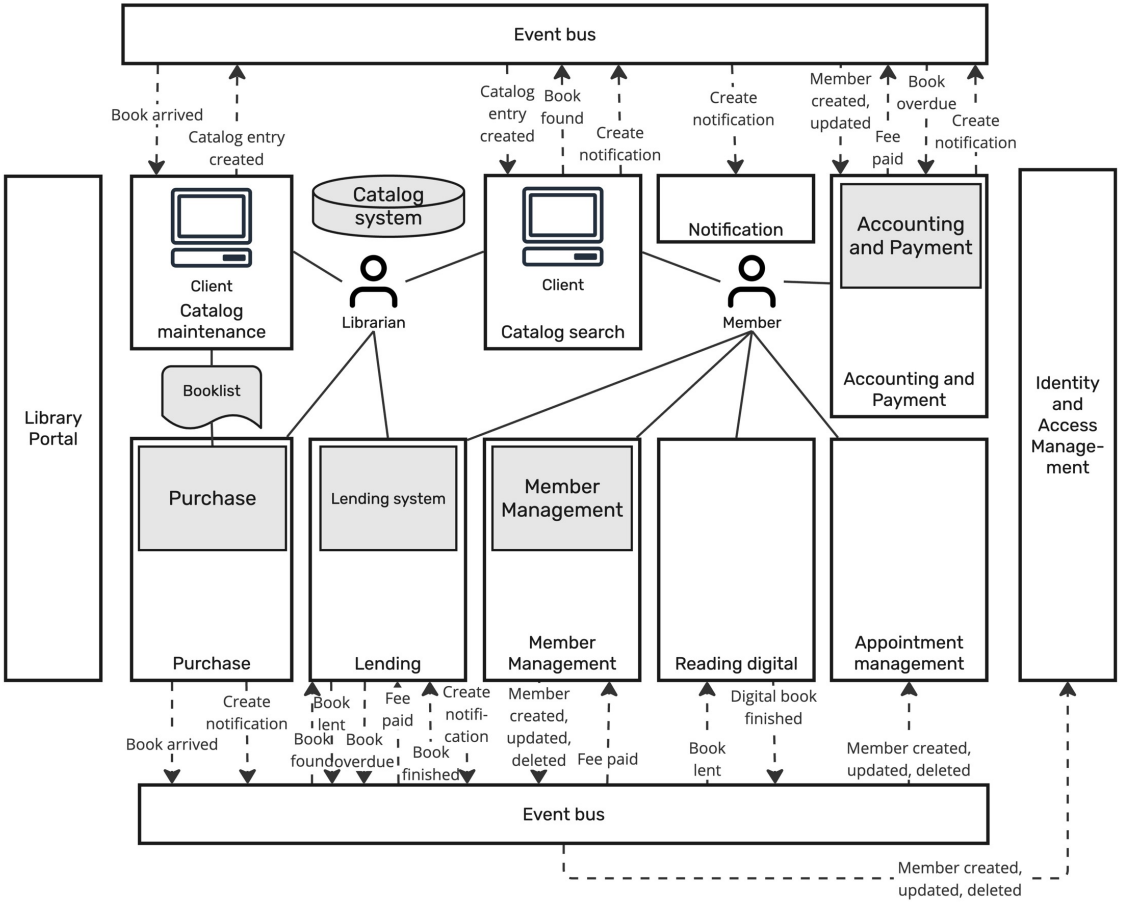


Analyse zukünftiges System

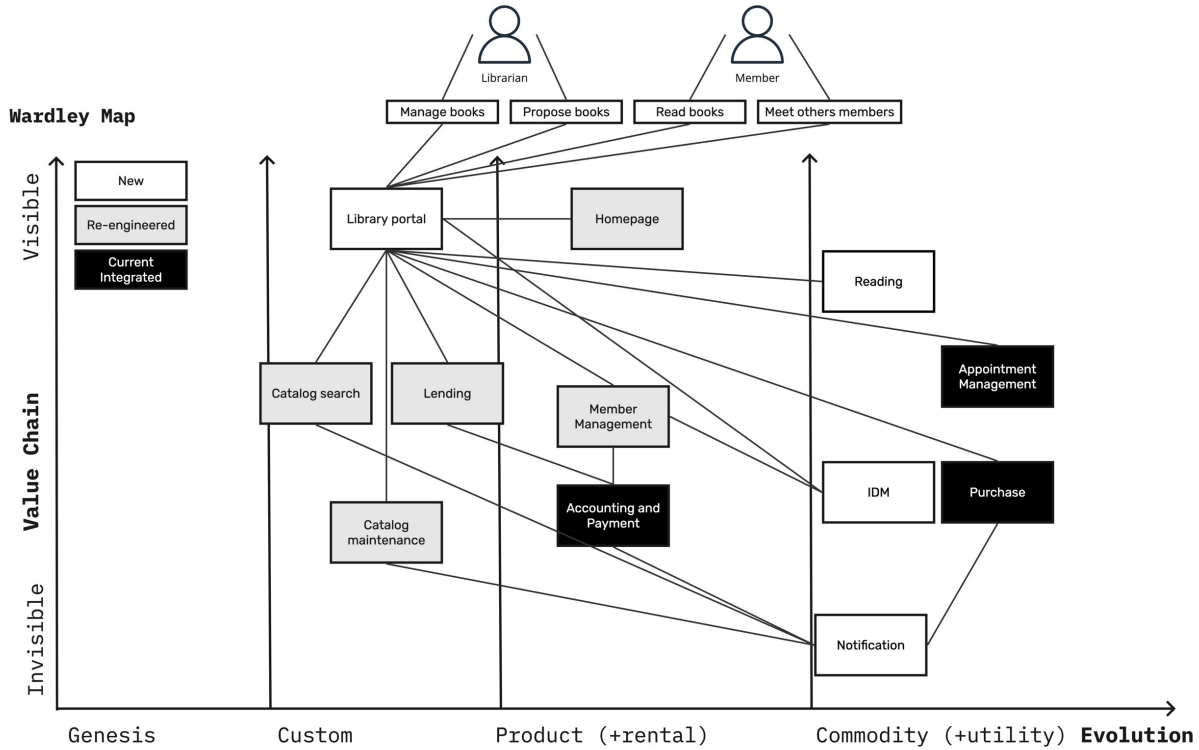




Library Architecture

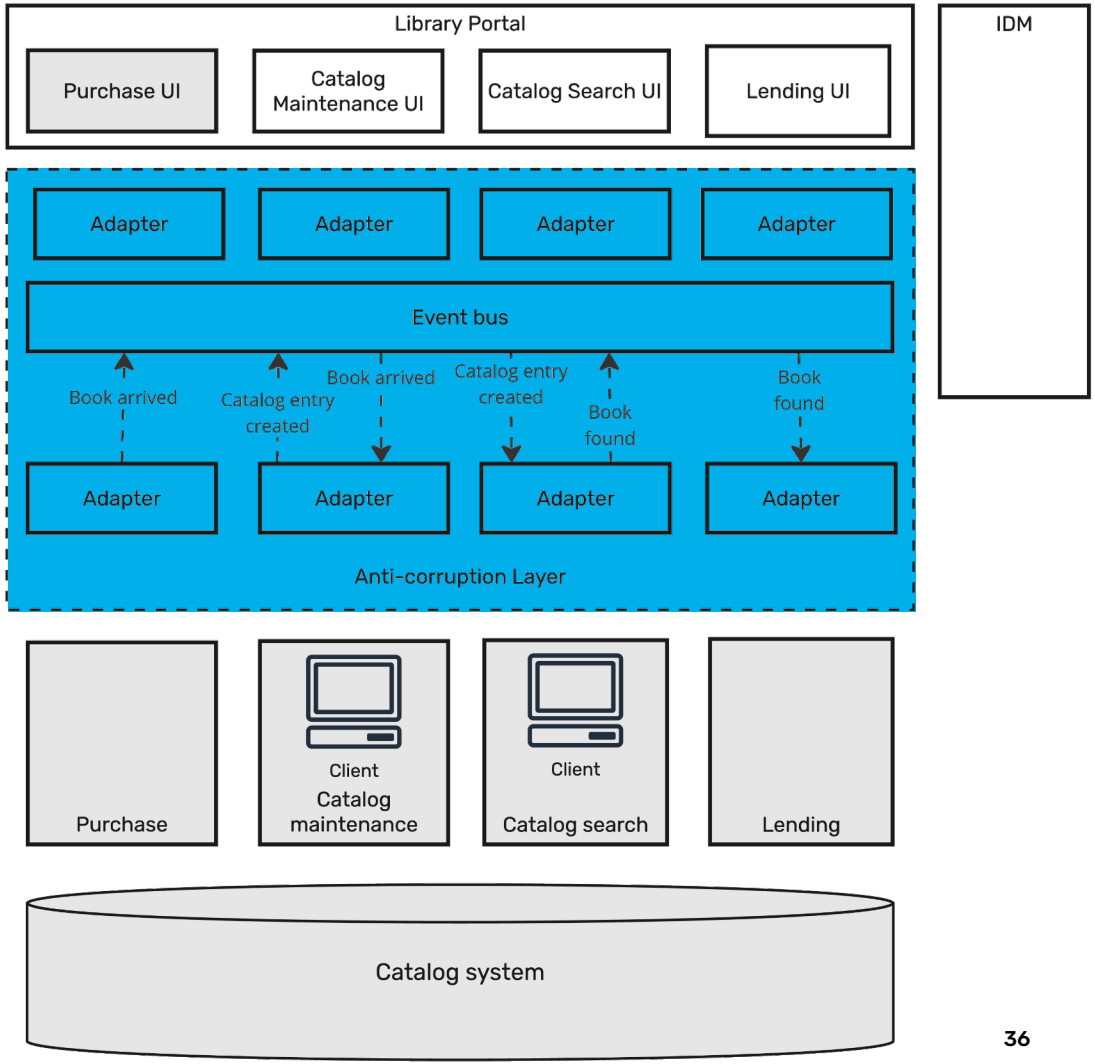


Priorisierung der Arbeiten
Wardley Map



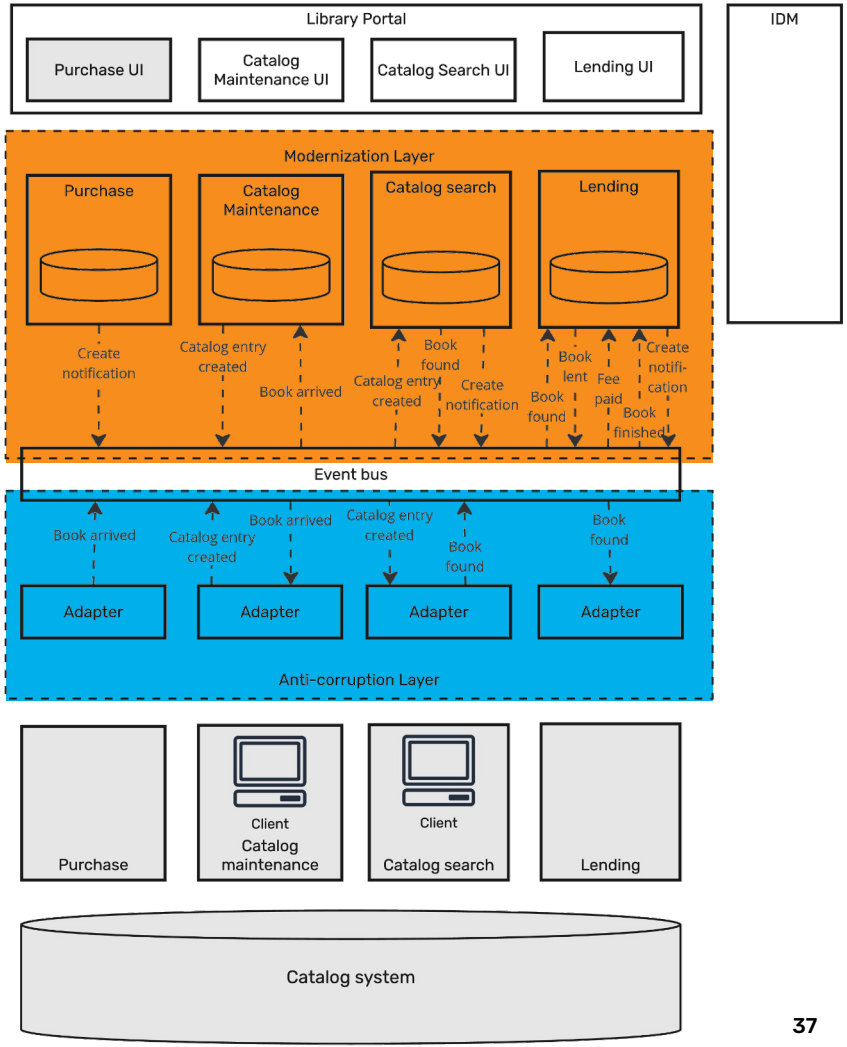
First Step – Anti-Corruption-Layer to the Legacy System

Anti-Corruption Layer schützt die neuen, modernisierten Teile vor dem alten Modell, das nicht mehr den neuen Ansprüchen genügt.

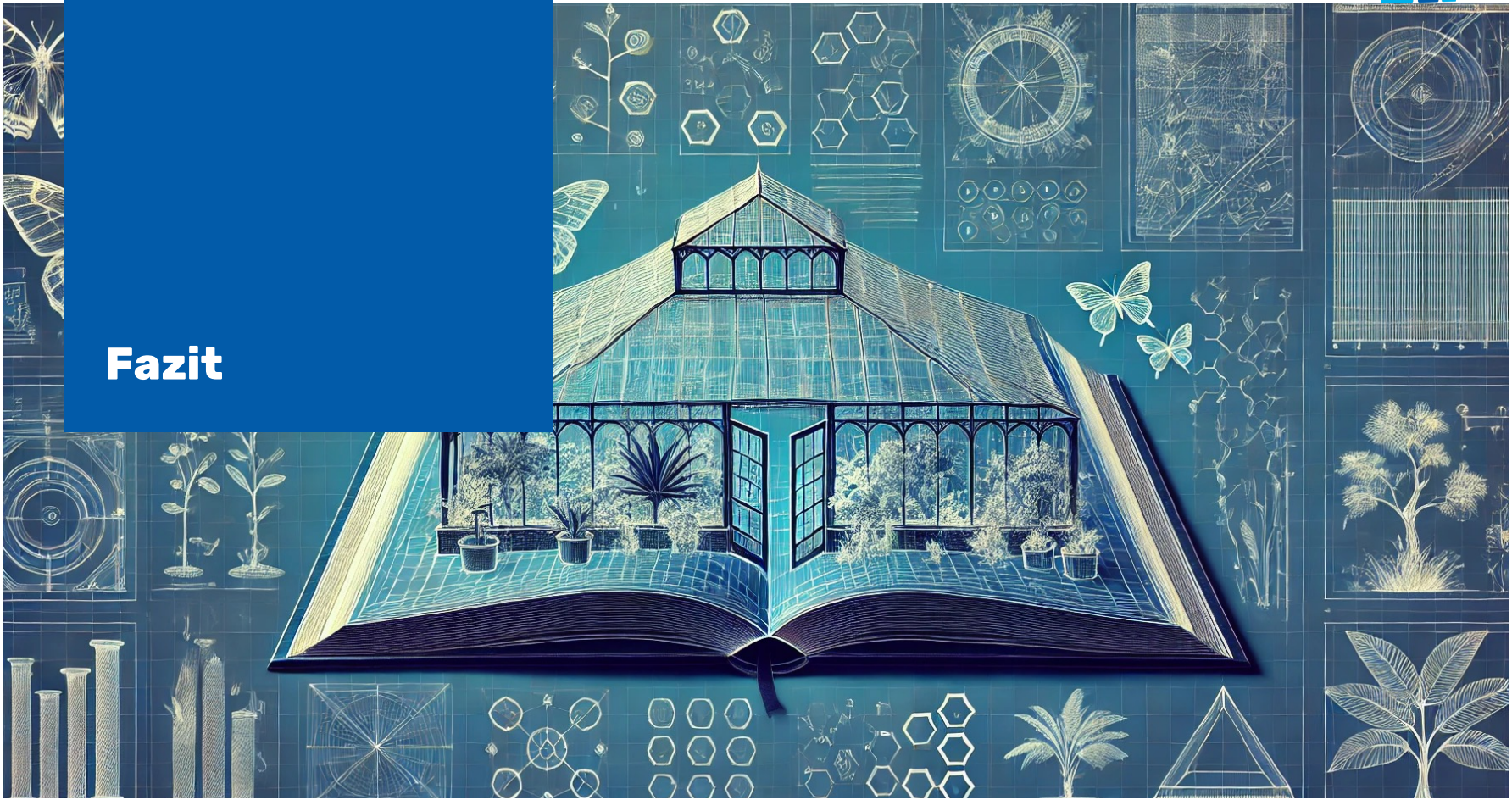


Second Step – Modernization Layer

Modernisierungs-Layer erlaubt Schritt-für-Schritt-Vorgehen beim Ablösen der einzelnen Funktionalitäten



Fazit



Fazit

- **Read the book *"Crafting Great APIs with Domain-Driven Design"*** ☺

- Domain-driven Design und API-Design gehören zusammen wie ein Paar alte Hausschuhe
- Das vorgestellte Vorgehen des **Synergetic Blueprints** kann sowohl auf der grünen Wiese als auch bei der Ablöse von bestehenden Systemen angewendet werden.

Kontakt Annegret
https://github.com/Grinseteddy
https://www.linkedin.com/in/dr-annegret-junker-141a99a4/

Kontakt Fabrizio
https://github.com/Lazzaretti
https://www.linkedin.com/in/fabrizio-lazzaretti/

Vielen Dank